Table of Contents

**Section 1 - Project Research Document**

Detailed Discussion2

Existing applications in this Domain2

Platform, Technology and Libraries4

Risks to the Project5

**Section 2 - Technical Application Architecture**

Use Cases7

Technical Architecture11

Software components11

Platform Libraries12

Distribution and Deployment13

Risks to the project14

Initial project prototypes15

**Section 3 - Post Development Documentation**

Installation Manual17

User Manual18

Post Project Review24

Project review24

Possible future extensions25

Review of Research, Analysis & Design phase25

Changes to class Diagram26

Final Class Diagrams27

Implementation Review29

Minutes of Meetings30

**Detailed discussion**

For my fourth year project I have proposed a mobile application which will offer a user an easy way of scheduling group and individual activities such as: social events, business meetings, appointments etc.

The application should support multiple users each having their own private events which cannot be viewed or accessed by other users of the application. On top of this, the application should provide the ability for users to invite other people to take part in events and to make specific events visible and accessible for other users to access and modify with their own comments, which will be visible to each user that is currently taking part in that event.

The application will require user validation in order to ensure that the correct people are accessing the correct events. there are a few methods of handling user validation; making a registration and log in system with the necessary security to ensure user data isn't susceptible to virtual threats and theft, Utilising login providers such as Facebook, Google+, Twitter and other social media platforms.

In order to provide this functionality, The application should consist of:

* A timetable to display the events the user had scheduled.
* Functionality to allow multiple users to have their own events separated from everyone else's events.
* Functionality to provide users a way to invite other users to view information on private events.
* Functionality to allow users to securely log in to the application to keep track of what events they have access to.
* A database to store the data pertaining to the events remotely (i.e. web server database).
* A web hosting provider to maintain and run the database for the application.
* A web API to service requests made by the application to the database and return request result from the database to the application.
* Functionality to provide to the user a map interface whereby they can place markers for whatever venue they have in mind for their event.
* A way to store this venue information and make it available for other users of the app to view.
* A method for users to comment on events they have been invited to and provide communication between users.

**Existing Applications in this domain**

App Name: Schedule Planner

Location: Google Play Store

Category: Productivity

This mobile app is a good representation of how an app designed to showcase a user's schedule should look and behave. The user interface this application uses is a good example of how to make the most use of the limited screen space of a mobile device while displaying as much relevant information to the user as possible.

This application shares some obvious similarities with my proposed project in that it has a timetable of events the user can create and have displayed to them in a schedule format.

Some differences however are; this app does not have the group event scheduling functionality, this app does not support the sharing of data between users regarding shared events. This app also does not have any kind of mapping functionality integrated into its timetable data where the app planned for the project will show the user the location of the event on a map.

App Name: Timetable

Location: Google Play

Category: Education

This app is another example of a lightweight, focused activity scheduling app with a focus on educational activities such as study planning and time management. this app's user interface has much less visual noise than the Schedule planner app and makes more use of the white space between Interface elements to convey the data it presents to the user.

This application is another good example of how to present a timetable application on a mobile device and it shares much of the functionality planned for my fourth year project in regards to organising and displaying data to users. However, it does not share the mapping or group vent scheduling and coordination functionality planned for the fourth year project.

App Name: Meetup

Location: Google Play

Category: Social

Meetup is a mobile application which allows users to view and attend organized social events such as hikes, book clubs, language lessons etc. This application makes finds exactly where you are located in the world depending on GPS data and is used to present the user nearby social happenings organized through Meetup.

A couple of this apps features are quite similar to what is planned for the fourth year project, with group event scheduling that people using the same app can organise and schedule events with one another with map support pinpointing exactly where the venue for the event is located. This application does not feature any timetabling functionality for scheduling events and organising them in a timetable format.

App Name: Yelp

Location: Google Play

Category: Travel & Local

Yelp is a mobile application which allows a user to search for shops, restaurants and services near their location. The app allows users to rank these venues from 1 to 5 stars as well as write a short review on the venue. This app features map functionality to present to the user the location of their desired venue.

This app is quite different in scope to the fourth year project however some of its key features can be considered similar to the project.

**Platform, Technology and Libraries**

Platform: Android, Windows Azure, ASP .NET MVC

Android

The mobile application portion of the project will be built using the Android SDK using the Android Studio development environment. Android suits the needs of this project well because there is plenty of built-in libraries to service http GET/PUT/POST/DELETE requests sent out over the internet and also plenty of methods of retrieving the data in different formats such as JSON or XML and parsing them into usable data.

Android also has good support for building the kind of Graphical User Interface we require for this project. Utilising XML files we can precisely specify how we want the User Interface to look and operate.

Windows Azure

Regarding web hosting, for this project we require a means of hosting a server that will serve as a meeting point for mobile device web clients to access the database which contains their schedule information

Windows Azure provides a web hosting service and platform for developing web based applications which can include a website, remote database, or a web API to service requests made to and from a web client. Windows Azure is very flexible because it supports any operating system, language, tool or framework which is very valuable for future scalability for the project. Windows Azure also is reliable, It offers a 99.95% SLA each month which roughly equates to about 5 minutes of downtime per month. This web service solution suits the needs of the project quite well and is made available for free using student licenses which is an added bonus.

ASP .NET MVC Web API

The mobile application requires a way to access and send data to be stored on the web server. To accomplish this we will implement a RESTful web API to handle http requests sent from our mobile application running on a phone or tablet. This API will be responsible for sending queries to our database on behalf of the mobile device and returning a result. The web API will be built using ASP .NET MVC and written in C#

Technology: Java, C#, JSON, MySQL, Entity Framework

Java

The mobile application will be built on the Android platform using the base Android SDK and using the Java programming language. XML will also be used alongside our Java code to build our app's user interface.

C#

The web API will be written in C# using ASP .NET

JSON

JSON will be the technology for transmitting data between our web server and our mobile application.

MySQL

A MySQL database will facilitate our data persistence because it is included in a Windows Azure web server and has support to easily maintain database tables and relationships between them through a User Interface built into the platform.

Entity Framework

Entity Framework is an Object-Relational Mapper which translates our model written in C# to SQL tables for our database.

Libraries: Google maps Android API, Android Native Libraries

Google Maps Android API

The Google Maps Android API will be how we will embed maps into our android activities. It will provide us with features such as 3D maps, location marking and longitude and latitude translation.

**Risks to the project**

Some Risks involved with developing this project include:

Libraries:

We are relying on Google Maps Android API support to facilitate the mapping functionality of the Application. This API will be our only means of displaying to the user the location of where events are taking place.

Windows Azure:

We are relying on Windows Azure to provide our web service to our application. Windows Azure offer a 99.95% SLA for uptime of its server's being available for use. However if it is not available for our mobile app to retrieve data then the application will be unable to support the group event scheduling and timetabling functionality.

We are relying on Windows Azure to also manage our data persistence using a MySQL database. if for any reason the web server should be unavailable then our app will not be able to communicate with the database and retrieve and store data.

Database:

Our data will be stored on a single database for this project. This is a rather large risk because if for any reason the database suffers from a fatal error which results in loss of our data, then the data the user's need for the app to function in regards to group event scheduling will be lost as well.

Android:

We are relying on the Android platform for this project to be built on. if there are any issues with versioning on older devices then it will require extra resources in order to correct the issue.

|  |
| --- |
| 4th Year Project |
| Technical Application Architecture |
| Christopher Newell X00084900 |

**1. Use Cases:**

**1.1 Title:** User creating an event.

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to create a new event.

**1.4 Level:**

**1.5 Story:** When the user wishes to create an event in order to store information about that event such as: venue, time, date, list of attendees. The user can add data relevant to that specific event and then create it by sending it to the database to be stored and available for retrieval.



**1.1 Title:** User retrieving list of events.

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to retrieve a list of events.

**1.4 Level:**

**1.5 Story:** When the user wishes to retrieve a list of events or a specific event, they set some search criteria which is then taken by the application and sent through the internet to the web API to be sent as a query to the database. the returned result set is then sent back to the user through the internet and displayed on screen, in the event a query fails due to the event not present in the database, the user will be notified.



**1.1 Title:** User logging into application.

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to log into application using social media login provider

**1.4 Level:**

**1.5 Story:** When the user wishes to log into the application to see what events they have created/are attending, they can log in through some social media login form as a form of security. In the event of success, the user logs in. In the event of failure, the user does not log in and is prompted again.



**1.1 Title:** User selecting a venue on map

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to select a venue on map

**1.4 Level:**

**1.5 Story:** When the user wishes to select a venue on map in order to specify exactly where the event is taking place and have it displayed as a marker on the map, the user opens the map on the mobile device and places a marker, then the positional data regarding that marker is taken and stored as an attribute of the event they have created.



**1.1 Title:** User signing up to event.

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to sign up to event.

**1.4 Level:**

**1.5 Story:** When the user wishes to sign up to event, The user selects an event and is shown the specifics of that event. The user can then decide to either sign up or if they already have signed up, delete their sign up



**1.1 Title:** User searching location on map

**1.2 Primary Actor:** Mobile User

**1.3 Scope:** Mobile User uses mobile application to search location on map and view events taking place at that location

**1.4 Level:**

**1.5 Story:** When the user wishes to search location on map and view events taking place at that location, The user enters as search criteria the location they wish to view events for. then the application will return a list of events for that location.



**2. Technical Architecture:**

2.1 Software Components

* + Android
  + Visual Studio
  + Android Studio
  + MySQL database
  + Windows Azure
  + ASP .NET MVC
  + Entity Framework
  + LINQ
  + JSON

**Android:**

The Android platform is an open source mobile device operating system that has become the most popular mobile device operating system in the world. Android is the chosen platform to build the mobile application for our project because it features a robust collection of native libraries which provide most of the essential functionality for our app such as; handling http requests and responses, providing a customizable Graphical User Interface as well as handling touch screen events such as tapping, sliding or pinching of GUI elements.

On top of this, there exists a vast library of externally developed, open source libraries for specific functionality such as displaying a map to a user and allowing interaction to take place such as marking locations and finding the distance between two points.

**Visual Studio:**

Microsoft Visual studio is a development tool which supports creation of projects built using ASP .NET MVC as well as integration with the Windows Azure web hosting platform. Using this tool, The creation of a web API to interact with our database will be much simpler to develop and will also be fully supportive of the technologies that we will be taking advantage of.

**Android Studio:**

Android Studio is an IDE specifically made for development on the Android platform. It is a freely available IDE and features Gradle-based build support, tools for Android specific refactoring, performance, version compatibility and built-in integration of many Google developed platforms and libraries.

**MySQL Database:**

MySQL is an open source relational database management system and is natively integrated as a component of the Windows Azure platform. This technology will be how the project handles data persistence and will have an interface in the form of a RESTful ASP .NET MVC web application. This interface will be how our mobile application will interact with the database.

**Windows Azure:**

Windows Azure is a web hosting service which comes bundled with a number of useful components to make the building of our web API easier. It will also provide us with hosting of a MySQL database which will allow us to access data over the internet using function calls to our RESTful web API built using C# and ASP .NET MVC. As an added benefit, there exists functionality to automatically publish the web API directly to Windows Azure through Visual Studio.

**ASP .NET MVC:**

In order to accept and respond to requests sent from the Mobile Application to the web server's database, a RESTful web API is required to handle this traffic. The web API will be built using the web application framework ASP.NET MVC, this framework is used to build web applications using the MVC design. The MVC programming model is the newer version of the previous Web Forms approach to developing web applications and is integrated and supported by Visual Studio.

**Entity Framework:**

Entity Framework is an open-source ORM (Object/Relational Mapping) framework which removes the requirement to write a ton of data access code which is used for programs to interact with databases (Create Read Update Delete operations). Entity Framework will automate much of the database code we will be required to write otherwise.

**LINQ:**

LINQ is a component of the Microsoft .NET Framework. It will provide a general purpose querying mechanism to our web API for specific requests sent to the database that will need to return result sets of data to the mobile application. LINQ is integrated into .NET languages such as C# so we will be able to write the necessary functions in C# to retrieve these result sets of data using LINQ.

**JSON:**

JSON is a lightweight standard for interchanging data across the internet. JSON is a language independent format which makes it excellent for sending data to multiple different mobile platforms such as Windows Phone and iOS devices. JSON uses attribute-value pairs to organise data sent and because of this, it is a much more lightweight and cost-effective tool to transmit data in comparison to another data exchange format such as XML which makes use of tags and bodies to send data.

2.2 Platform Libraries

* + Google Maps API for Android V2
  + Java (Android SDK)
  + C# (.NET MVC)

**Google Maps API for Android:**

Google Maps Android API V2 is a library which allows developers of Android applications to embed maps into an activity.

These maps offer a variety of features such as 3D maps, indoor maps, satellite maps of large stretches of land, user customization of a map by placing markers to indicate places and points of interest.

This library also works with a large variety of related APIs such as the Places API which access a database of over 100 million points of interest, the Location APIs which can allow a mobile application to be location-aware (i.e. the mobile application will be able to pinpoint the location of the device it's running on in the world).

**Java(Android SDK):**

The Android SDK is a software development kit that allows developers to create applications for the Android platform. This platform includes tools such as; Sample projects to explain pieces of functionality, development tools to create and run Android projects, libraries of core functionality to build Android applications for a mobile device, a device emulator for testing a mobile Android application on a wide range of devices with different system specifications and hardware configurations.

Android applications are written in the Java programming language which means that Java will be the language chosen to build the mobile application for the project.

**C#(.NET MVC):**

As outlined above in the Software Components section of this Technical Application Architecture document, The web service or application which will service requests to and from the MySQL database will be built using ASP .NET MVC and written in the .NET language C#. This will allow the project to take full advantage of technologies and support featured in the .NET framework for building MVC applications and hosting them through a web server.

2.3 Distribution and Deployment

* Cloud Database with RESTful web API
* Log in through social media

**Cloud Database with RESTful web API**

This project will make use of a MySQL database hosted on a Microsoft Azure Web Server. This database will be how the project maintains data persistence. The database will be required to store, update , delete and retrieve data for our mobile application.

Mobile Devices have limited system resources, especially when internet requests are involved. Because of this limitation, opening up and maintaining a continuous session with the database is a bad idea due to a number of reasons:

* + maintaining sessions from a large, distributed collection of clients over the internet will be a very large load for the database to manage.
  + Mobile devices are likely to simply time out of the session and shut it down if the request/response/authentication takes any longer than a couple of seconds
  + The number of mobile platforms will be very limited, each requiring a connecting class for each system's programming language to set up the session to the database
  + Security becomes an issue when you have multiple different platforms each using its own language to request data from the database

Because of these reasons, we require a different approach to service requests sent over the internet to our database and to send results back to our mobile application.

A RESTful implementation of a web API will act as the mediator between the database and the mobile application. This API will make use of simple Http requests to different URLs located in our web API. These URLs will be tied to functions which then act on the data in the database and return the data back to our application.

This approach does away with having to maintain a session to the database with the mobile application and also opens up the application to be ported to other platforms such as iOS or Windows Phone as there is no platform-specific code involved. The only thing the device will need to access the database is the ability to make Http requests which if they have a browser installed, is a likely scenario.

**Logging in through social media websites**

Managing User Authentication is a critically important area of mobile development. If it is handled poorly or oversights are allowed to persist, the user's data or device itself could be at risk of compromise or attack.

Rather than risk a security flaw in building a user authentication system from scratch, a social media service will provide the log in operation which will then provide the mobile application with a means of user authentication for this project.

there are plenty of different authentication providers such as Twitter, Facebook and Google+. each has support for adding a log in button to the application which will then ask the user to provide their account details for each of those social media platforms.

Managing user authentication this way will save a lot of development time that would otherwise be spent on building an authentication service which may end up becoming vulnerable and insecure. Including the ability to log in with a social media account and then using the access token that creates to authenticate users is a much simpler solution.

2.4 Risks

* Windows Azure being unavailable
* Google Maps being unavailable
* Level of experience with technologies involved with the project

**Windows Azure Unavailability:**

The Windows Azure web hosting platform is this projects only means of servicing requests made through the internet from the project's mobile application to the database. In fact, the database itself will be contained entirely on their web servers and will be inaccessible in the event of the Windows Azure service becoming unavailable for whatever reason.

This also means that once the subscription to Windows Azure runs out, both the web service, the MySQL database and the data pertaining to the mobile device's users will be inaccessible until the subscription is renewed.

**Google Maps Unavailability:**

The only library the project uses to display maps and provide map customisation functionality to the user of the mobile application will come from the Google Maps API for Android V2 library.

If, for whatever reason this service becomes unavailable then the mobile application will be unable to display maps to the users, provide map customization features such as marker placement and location searching, any other APIs which are used to provide additional functionality such as the Places API or Locations API will also be affected by this.

**Inexperience with technologies involved with the project:**

This project makes use of a variety of technologies which are used to create web based applications. each of these technologies will require some degree of revision or practice to use effectively for the project. In the event that a piece of functionality does not behave as intended or there is uncertainty on how to achieve some functionality, this scenario could cause the project to run into some extra difficulties which require more work to resolve.

**3. Prototype**

3.1 Prototype Deliverable for week 8

**Use Cases to be delivered:**

1.User Creating an Event

2. User retrieving list of events

3.User signing up to event

3.1 Testing Strategy

The Testing Strategy for this iteration will be focused on building the web API and Android application and setting up some of the basic functionality that the project will require.

In the Android application's case, the usage of emulators included in the Android SDK to test different versions of Android will be very useful to ensure that the application can run on as wide a variety of devices and versions of Android as possible.

For the web application, the use of Unit Tests for the different query functions the API will possess and dummy data to simulate a database connection will initially allow us to test that communication between the mobile device and the web API through the internet using Http requests is working and that the correct data is being sent and retrieved through the internet.

This will also allow us to test how we will transmit the data using JSON and how each end of the project will handle parsing this data and representing it correctly.

3.2 Prototype Deliverable for week 11

**Use Cases to be delivered:**

1. User logging into app

2. User selecting venue on map

3. User searching location on map

3.2 Testing Strategy

The testing strategy for this prototype will be focused on enabling Location based data using the Google maps API and how that data will be taken from the user input on the map and saved as an attribute of an event. The web API will continue to make use of Unit Tests to ensure that no new code introduced to the API will have an adverse effect on how the older functions operate. For the Android application, emulators for different devices and versions of Android will be used to continue testing changes and updates to the project

|  |
| --- |
| 4th Year Project |
| Section 3: Post Development Documentation |
| Christopher Newell X00084900 |

**1. INSTALLATION MANUAL**

The mobile Application installs and is fully supported on Any Android device with a operating system version higher than 4.0.0.

Features of the application which are supported by Android 4.0.0:

* Calendar Provider
* Latest Version of Google Play Service Pack
* Windows Azure SDK for Android

The android application portion of the project can be installed on any android device whose Android API level is greater than version 16 (Android 4.0).

The web service is built to use a Windows Azure Notifications Hub which is tied to a Google license key for the Google Cloud Messaging for Android library to handle push requests sent from the web API to devices that the Android project is installed on. Push notifications sent to the Android applications from the web API will still function on any web hosting solution other than Azure.

However, extra steps must be taken to ensure that the Android application correctly sends its device ID and then receives a registration key from the Google Cloud Messaging service if the project is moved to any other web hosting solution other than Windows Azure.

The Web API project makes use of many .NET Framework technologies such as LINQ and Entity Framework for database creation and data manipulation so the project should still function correctly with many of the most common database OLTP systems in use in enterprise applications today.

That being said, the project was developed and built using Windows Azure as the web hosting service so it is recommended to use Azure as the project's web hosting service provider when installing/deploying the project.

**2. USER MANUAL**

The mobile application is comprised of several UI screens each with a specific purpose. Navigation through the mobile application

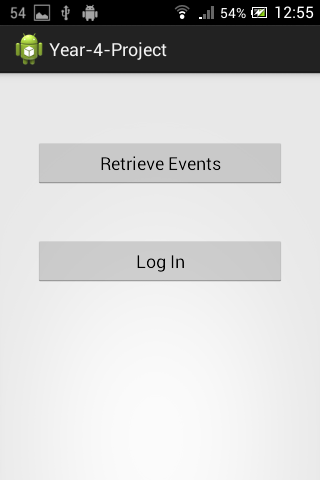
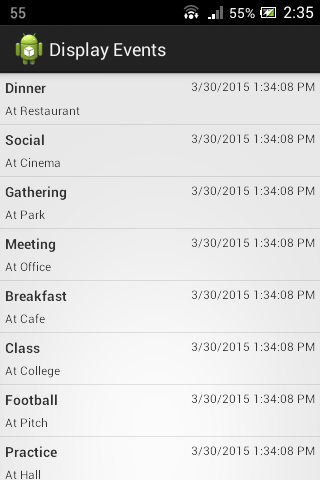
 

Figure 1.1 - Main Activity Figure 1.2 - Display Events Activity

(Fig 1.1)

The first Screen the user is presented with allows them a choice between two options, "Retrieve Events" and "Log In" which will take them to other activities within the application.

(Fig 1.2)

This Activity displays a list of every activity currently stored in the project's database. This data is accessed through the project's RESTFUL web API.

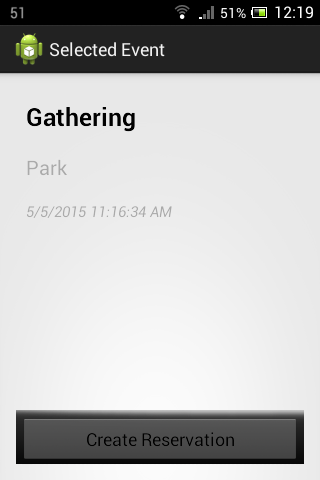
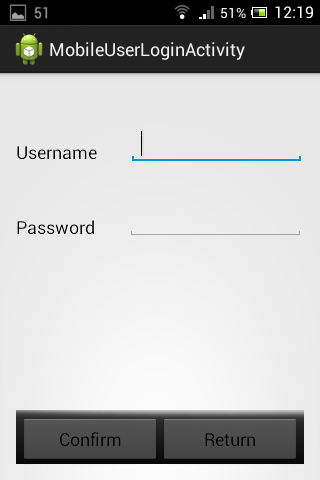
 

Figure 1.2a - Single Event Details Activity Figure 1.3 - Log In Activity

(Fig 1.2a)

This Activity starts when a user selects an event by tapping on their selection on the previous activity(Fig. 1.2). This Activity opens the event on a new screen and then allows the user to create a reservation for that event by pressing the "Create Reservation" Button. This button prompts the user to log in by opening the Log In Activity and then upon successful log-in, assigns this event to their event reservation list.

(Fig 1.3)

This activity lets the user provide their log in credentials and log into the mobile application from the Main Activity(Fig.1.1) and the Single Event Details Activity(Fig 1.2a). If logging in from the Main Activity it will bring the user to the logged in menu(Fig. 1.4) and if logging in from the Single Event Details Activity it will create a new reservation for that user's account.

If the user's login fails, either from incorrect information or a failure to input information, a toast message appears asking the user to log in again.

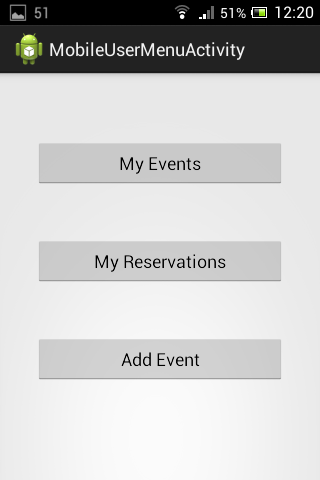
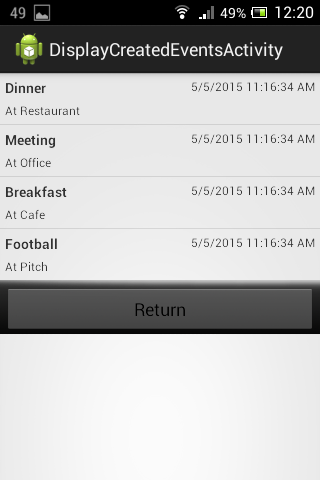
 

Figure 1.4 - Mobile User Menu Activity Figure 1.5 Display Created Events Activity

(Fig 1.4)

This Activity displays a menu to the user once they have successfully logged into the application and allows them to perform actions on their personal list of created events, and their list of reservations to other user's events. Users can also create a new event and have it added to their list of owned events from this menu.

(Fig 1.5)

This Activity is where the user can see the events they have created and from here, they can select an event to open the Single Event Details Activity (Figure 1.2a) except now they have the option of editing or deleting the selected event rather than placing a new reservation for the event.

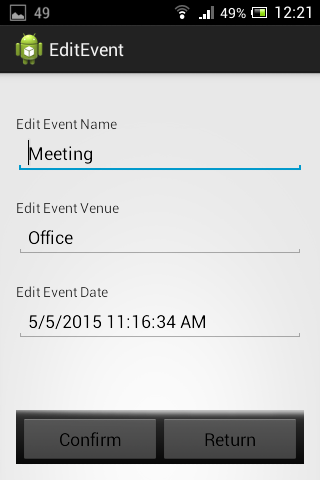
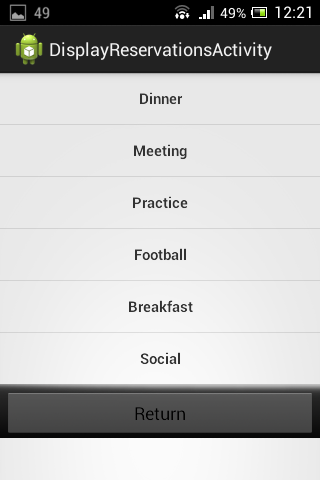
 

Figure 1.6 Edit Event Activity Figure 1.7 Display Reservations Activity

(Fig 1.6)

This Activity allows the user to make changes to and update an event stored in the database that they have created. A user can get to this activity from the Display Created Events Activity and by selecting an event by tapping on it.

(Fig 1.7)

This Activity displays a list of reservations the user has signed themselves up for participation in by using the Activity in Figure 1.2a.

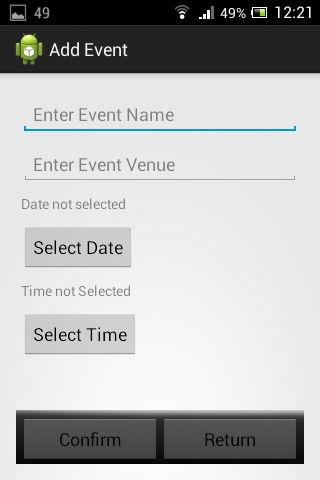


Figure 1.8a Add Event Activity

This Activity allows a user to create a new event for people to create a reservation for by entering in a event name, venue and time and date. selecting either the "Select Date" or "Select Time" buttons will open a new activity overlay with a picker for the user to make a selection (seen in Fig 1.8b). Once they have entered in their details the event data is sent to the project's web API which then inserts the new event into the project's database. The activity then passes this data to the Android Calendar Application

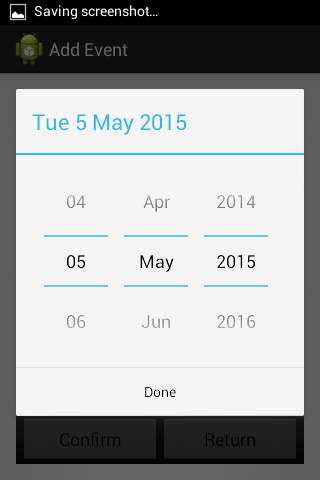
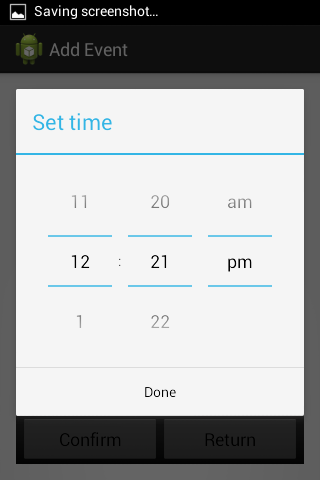


Figure 1.8b Time and Date Picker Activities

These Activities are a part of the Add Event Activity(Fig 1.8a) user interface and allow the user to make a selection of a time and a date for their newly created event

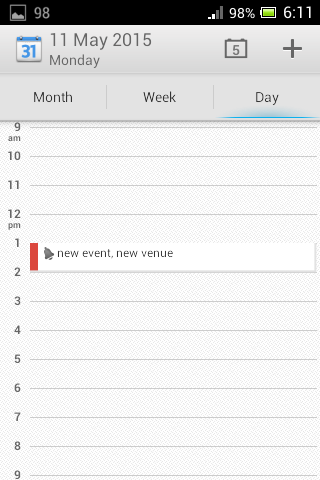
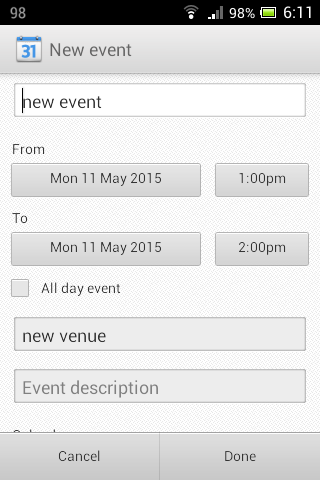


Figure 1.9 Calendar Activity

This is a native Android application which is invoked once a user creates a new event using the project's mobile application. These details are sent to:

1. The projects Web API for storage and to mark the logged in user as the creator of the event(Figure 1.4)

2. The Calendar application using the Calendar Provider Android library. The input fields in the Calendar application are pre-filled with the details of the newly created event.

**3. POST-PROJECT REVIEW**

Project Review

The Components of the project are as follows:

* The Android Application
* The Web API
* The Database
* The Web Hosting Platform

The Android Application portion of the project handles the displaying of data (i.e. all user events, created user events, user reservations) and also performs the role of facilitating user input to interact with the database through the RESTFUL web service. The components that make up the entire android application are:

Azure SDK for handling of push notifications from an Azure Notifications Hub. The Google cloud Messaging Library for Android to receive, register and accept and display these push notifications correctly on the user's device. Native Android libraries are used to provide the user interface on the phone and to create user input and data display components on the project's GUI.

The web hosting service for the project, which provides hosting of the project's database and web API project is Windows Azure, this web hosting platform offers benefits beyond the base requirement of web hosting. Visual Studio, which was used to create the web API project, has many tools to streamline the process of publishing a new build of the web API project straight to Azure from the IDE. Windows Azure also has the benefit of providing a Notification Hub component which supports and handles the Google Cloud Messaging service which is used in the project to send push notifications to the Android application.

The database for the project was mapped out, created and managed by using Entity Framework Code First Migrations. This approach allowed greater flexibility when changes were made to the project's models and then reflected in the database by creating a new Migration file and then editing the seed data used to populate the database with test data. Again Visual Studio supported the Entity Framework library as it is a part of the .NET Framework.

The web API portion of the project exposes a set of URLs for other applications to consume. It's role is to handle HTTP requests sent to the web service and then query and return or modify data in the project database depending on which function was called. It does this by mapping a set of URLs each of which is assigned to a specific function in one of the controller classes. This web API is built using Microsoft's .NET Framework and is makes use of various web communication technologies to support RESTful web APIs and handle data querying and storage such as Entity Framework, ASP .NET Web API Project template, LINQ and it was developed using Visual Studio.

The ASP .NET Web API portion of the project uses LINQ as its data querying component to provide the functions which are queried through the RESTFUL Web API from the project's mobile application component. These LINQ Queries are performed once a request is made to one of the functions in the web service once a request is made to a URL that the function is mapped to using HTTP verbs GET, PUT, POST and DELETE.

Possible extensions of the project

Google maps functionality

The initial project functionality outline was to include Google Maps support for selecting and displaying longitude and latitude information of event venues on Google Maps. This feature was replaced with the inclusion of push notification support midway through the project which was not initially considered but discovered to be a more important feature when considering usability. A future extension of the project could be to include this Google Maps support.

User accounts creation and deletion,

A possible extension of the project would be to expand the user accounts system to allow the creation of new user accounts

Contact list of specific "Friend" users

A natural expansion of the project would be to include some functionality in order for users to more easily come into contact with one another. A feature that would solve this would be to allow users to create a list of friends within the app, this could be accomplished by storing this list of friends as a list of usernames per user account or to make use of some external social media account to provide this functionality.

A social media platform such as Facebook could also provide the ability to synchronize a user's event schedule with their calendar on their Facebook account and to also provide another means of notifying users of changes to events they have signed up to. Posts concerning event details could also be made to the user's wall in order to garner more interest in the event from other users.

Other social media platforms such as Twitter or Google+ could be used for a similar purpose and another potential expansion of the project would be to allow users to log into the application using a social media account that they own. This would also provide the added benefit of allowing those social media platforms to provide security for a user logging in to the application.

Review of Research, Analysis and Design phases:

Review of Implementation phase:

The implementation phase for the project caused a shift in priorities for functionality of the project. The initial plan was to include support for Google Maps so that a user could select a venue for their event on a real map and have the latitude and longitude information stored to be passed to other devices viewing the event.

This was replaced with the inclusion of push notification integration into the project which was not initially considered for the project but became a necessity for project usability. Users needed to be informed when changes were made to an event that they are signed up to(i.e. Deletion, edit of venue/time/date). The decision to shift priorities was made for this reason: if this functionality was not included in the project it would have been a major usability flaw compared to the pleasant but not as important usability-wise feature of having a map displayed on screen

Additional Research for this project included:

* Implementation of Google Cloud Messaging for Android.
* Implementation of Azure SDK for Android

Preparing the project to support push notifications to users by using the Google Cloud Messaging library for Android was the first extra research topic required. the purpose of implementing this functionality was to allow the users of the mobile application to receive updates on status changes to events, specifically when an event has it's information changed by the event creator either when it is edited or deleted.

This library requires a new license key to be generated via Google Developers, the storage and management of device keys to identify users with the project mobile application installed to send these push notifications to the users upon a web request made to the Web API.

The next extra research topic was the implementation and creation of a Notification Hub on Windows Azure to handle the push notifications that are sent from the Web API as well as Implementing the Azure SDK for Android to actually receive and handle these push notifications by displaying them in the notifications menu on the Android device.

Changes to Class Diagrams & Data Model

During a project meeting with the panel it became apparent that there was an issue with how all mobile users have the same level of privileges concerning the creating, editing and deletion of events (i.e. any user could access/delete the data for any event). This issue led to a major change of the relationships shared between the classes used for data modelling of the database for the project.

This change of relationships was the inclusion of an entirely new class called Reservations which acted as a link between a specific user and a specific event, this enabled the project to keep a record of which users where signed up to a specific event as an attendee and then a new property was added to the event model class to specify which user was the creator (Identified by the user's email attached to their account).

This new property allowed the project to retrieve a set of the events that a user had administration over once they had logged in using the Android application which would populate a screen that the user could choose between performing edits or deletions to the database using a menu.

On top of the changes made to the existing set of models and tables, another issue was created when attempting to serialize the data from the database into a format suited to data transfer over the internet, for example JSON. This issue appeared due to the addition of the navigation properties mapping a one-to-many relationship between tables in the database which created a circular reference. The change made to the Class structure of the web API project was to include new classes which did not generate or map to any database tables but rather were used to create copies of objects from the database, minus the navigation property field.

Final Class Diagrams

Final Class Diagram for Android Application



Final Class Diagram for web API



How the implementation went, problems encountered and solutions proposed

Informing of users of changes made to event details - push notifications

The first major change to the overall functionality of the project was to replace Google Maps support with support for Google Cloud Messaging to allow push notifications to be sent from the web service to each Android application client registered with the project. This decision was made to improve usability of the project as there was no way previously to inform users of changes made to event they had signed up to that were managed by other users.

The implementation of push notifications was a success, the time between researching a solution and finding and implementing Google Cloud Messaging into both the Android application project and the web API using a Windows Azure Notification Hub was a straightforward and well documented process which did not add much extra complexity to existing project code. all that was required was the addition of extra classes to each project to support this functionality.

Overall there were few problems encountered when developing the project and solutions were readily available as these issues were commonly encountered problems with similar projects to this one.

The most important of these problems were:

1. Changes had to be made to database tables in order to correctly model relationships between the data objects. This change came in the form of the reservations table being added and subsequently navigation properties included in Events and MobileUsers models to this new Reservations model.

2.Data transfer objects were then required as these new navigation properties to the database models could not serialize correctly as JSON objects.

* Changes made to database tables - reservations table added, navigation properties included in Events and MobileUsers models

The inclusion of navigation properties added to both the Events and MobileUsers models and their respective database tables initially created an issue with serialization of the data into JSON format for data transfer between the web API and the Android application.

The One to Many relationship between Events and the newly created Reservations table and likewise for MobileUsers had the unintended effect of creating a circular reference between these models once an attempt was made to serialize these objects and as a result, caused an exception to be thrown whenever a request was made to retrieve any of these objects from the database.

* Data transfer objects for each class

This issue was solved by the creation of a Data Transfer Object for each class (named: EventDTO, ReservationDTO and MobileUserDTO) which did not contain a navigation property, was not a data model to be mapped to a database table and whose only purpose was to create a copy of the retrieved object excluding data retrieved in the navigation property, serializing that object instead in JSON format and then returning it as part of the web API functions. This approach circumvented the issue of an exception thrown due to a circular reference and allowed the data to be returned as part of a web service request correctly.

Minutes of Meetings

**Meeting 1:** This meeting was mostly a review of the project proposal and a discussion on where to begin work for the first iteration of the project.

**Meeting 2:** Basic data retrieval from a web database to an example Android application was demonstrated. A web database with some test seed data was created and used for these tests rather than a collection of object in-memory on the web server.

**Meeting 3:** An expansion of the RESTFUL web API was shown, the project could now support adding, updating and deleting of objects to the database, Reservations and Mobile Users models were then added to support event ownership and editing/deleting privileges. The Android application was expanded to allow users to view objects retrieved on a new menu.

Discussion was focused on how users would find out about changes made to event objects from the database and a few solutions were discussed, either through the use of emails sent out to users from the web API to their email addresses or to integrate push notification handling on the Android application & web API.

**Meeting 4:** Demonstration of push notification functionality to the Android application from the web service using Google Cloud Messaging library for Android and a Notifications Hub on Windows Azure. Android application user interface was greatly expanded to support user account log in, user specific event creation, editing and deleting operations using the Android application. Discussion was focused on user interface design and suggested that the native Android calendar application be integrated to improve user experience.

**Meeting 5:** Demonstration full application up to this point including demonstration of new functionality of adding event objects to the native Android calendar upon creation of a new event or creation of a new reservation to another user's event.